High-dimensional data exhibit **concentration of measure** and **curse of dimensionality**.

Concentration of measure: for high-dimensional objects, most of the volume is concentrated near the surface. Consider any object A in $\mathbb{R}^d$. Shrink A by some small amount ϵ to produce a new object $(1-\epsilon)A$. It is the case that $Vol((1-\epsilon)A) = (1-\epsilon)^d Vol(A)$. It also is the case that for any x, $1-x \leq e^{-x}$. So, $\frac{Vol((1-\epsilon)A)}{Vol(A)} = (1-\epsilon)^d \leq e^{-\epsilon d}$. As $d \to \infty$, then $(1-\epsilon)^d \to 0$. This implies that nearly all the volume of A is contained in the portion of A not included in $(1-\epsilon)A$.

Curse of dimensionality: because of concentration of measure, some nice things we would otherwise like to do become intractable. As dimensionality grows and concentration of measure takes over, two random elements are likely to be very far apart. This means that nearest neighbor methods don't work well.

Kernel density estimation (KDE): As motivation, suppose that we have some d-dimensional data $\mathbf{X}$ and class labels $Y = k_1, k_2...$, and want to predict Y based on $\mathbf{X}$. In other words, we're considering $P(Y = k|X = x)$, which we know by Bayes is equal to $\frac{P(X=x|Y=k)P(Y=k)}{P(X=x)}$. We don't particularly worry about the bottom part of that equation since it doesn't depend on Y and becaues we often have decent information available for it. We often have good information for the prior $P(Y = k)$ or can just use the sample proportions. So the chief challenge is estimating the the density function $f_k(x)$ associated with $P(X = x|Y = k)$. We have two ways of doing this: parametric (assuming some model, e.g. Normal, and estimating parameters for it via MLE) or non-parametric (make fewer assumptions).

KDE is a non-parametric method for $\hat{f}(x)$. As the simplest possible example, we could use a histogram, where we divide the range of observed x's into some number h of bins; count the proportion $\hat{p}_i$ of observations in bin i and divide by the length of the bin h. In this case, $\hat{f}_h(x) = \sum_i^m \frac{\hat{p}}{h}$ (the subscript is to emphasize the degree to which this estimator depends on choice of h). But, this is not a great estimator because observations very close to the border only contribute to the density estimate for one bin, when it seems like they should contribute to both. So, we use some function to weight how much an observation should contribute to estimation of density at some point, depending on how far away the observation is from the point (e.g., observation $x = 100$ should not much increase our estimated density at $x = 1$, but should contribute a lot to our estimation of density at $x = 101$). This weighting function is called a *kernel*. Implemented in one dimension, we have: $\hat{f}_{kde}(x) = \frac{1}{n}\sum_{i=1}^n \frac{1}{h} K(\frac{X_i - x}{h})$. In multiple dimensions, we have: $\hat{f}_{kde}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^n \frac{1}{|H|^{1/2}} K(\frac{\mathbf{X_i} - \mathbf{x}}{h})$

Our main requirements are that the kernel integrate to one, and that it be symmetric. The most common are Epanechnikov and normal (Gaussian). Our big choice is again the size of window to consider (e.g., the standard deviation in our Gaussian kernel). A large window means high bias and low variance (under-fit). A small window means low bias and high variance (over-fit). A method of evaluating candidate kernels is the **mean integrated squared error (MISE)**, aka **L2 risk function**, which is: $E(\hat{f} - f)^2 = \int_{-\infty}^{\infty} (\hat{f}(x) - f(x))^2$. The optimal kernel might be one minimizing this loss function.

The problem is that as d grows, the performance of KDE suffers, and there is no estimator achieving better performance. The fundamental issue is that the CoD means points become sparsely distributed in space and so h has to become very large to ensure that enough points fall into each neighborhood.

Naive Bayes: as a motivating example of why KDE is useful, however, we consider Naive Bayes. We make a big assumption that given a particular label $Y = k$, the p features $\mathbf{X}_k$ are independent of each other, implying that the joint density of $\mathbf{X}_k$ is just the product of the individual densities: $f(\mathbf{X}_k) = \prod_{j=1}^p f(\mathbf{X}_{jk})$. The nice thing is that we can estimate these individual densities via KDE (since we're doing them one at a time the CoD doesn't apply). So, we can end up with $P(Y_i = k|\mathbf{X}_i = \mathbf{x}) = P(Y_i = k)P(\mathbf{X}_i = \mathbf{x}|Y_i = k) = \hat{p}_Y(k)\hat{f}_{X|Y}(\mathbf{x}|k)$, where $\hat{p}_Y(k)$ is our prior, $\hat{f}_{\mathbf{X}|Y}(\mathbf{x}|k)$ is our KDE-based estimate of the likelihood of vector $\mathbf{x}$, and we don't care about the constant denominator that does not depend on Y. We use the **maximum a posteriori (MAP)** decision rule, which means we chose the $\hat{y}$ that maximizes the aforementioned function:

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, \hat{p}_Y(k)\hat{f}_{X|Y}(\mathbf{x}|k)$$

This is a useful technique when the dimensionality is very high (e.g. NLP, spam detection, etc.) in part because of the ease of computing KDEs.